Principal Component Analysis in CGAL

Ankit Gupta, Pierre Alliez, Sylvain Pion

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Principal Component Analysis in CGAL

Ankit Gupta — Pierre Alliez — Sylvain Pion

N° 6642

Septembre 2008

Thème SYM

ISSN 0249-6399 ISRN INRIA/RR--6642--FR+ENG

Principal Component Analysis in CGAL

Ankit Gupta* , Pierre Alliez† , Sylvain Pion‡

Thème SYM — Systèmes symboliques
Projet Geometrica

Rapport de recherche n° 6642 — Septembre 2008 — 13 pages

Abstract: Principal component analysis is a basic component of many geometric computing and processing algorithms. It is most commonly used on point sets, although applicable as well to sets of arbitrary primitives through the computation of covariance matrices. In this paper we provide closed form formulas of covariance matrices for sets of 2D and 3D geometric primitives such as segments, circles, triangles, iso rectangles, spheres, tetrahedra and iso cuboids. We also describe the method of deriving covariance matrices for their dimensional variants such as disks, balls etc. We finally discuss the flexibility and added value of the present approach by discussing its potential use in applications. Our implementation will be available through the next release of the CGAL library.

Key-words: Principal Component Analysis, geometric primitives.

* IT Bombay
† INRIA
‡ INRIA

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Analyse en composante principale dans CGAL

Résumé : L'analyse en composantes principales est un outil de base pour le calcul géométrique et le traitement numérique de la géométrie. On l'utilise pour estimer des normales à partir de nuages de points, pour calculer un tenseur d'inertie d'un solide, ou encore pour l'approximation de surfaces. Bien que l'analyse en composantes principales soit utilisée le plus souvent à partir de nuages de points, on peut l'appliquer à tout ensemble de primitives géométriques (sans les échantillonner au préalable) en calculant la matrice de covariance associée en forme close. On décrit dans cet article les formules des matrices de covariance d'ensembles de primitives géométriques 2D et 3D telles que des segments, cercles, disques, triangles, rectangles alignés avec les axes, sphères, boules, tétraèdres et parallélépipèdes alignés avec les axes. On discute la valeur ajoutée apportée par une telle approche ainsi que leur utilisation potentielle dans les applications. L'implantation associée sera disponible dans la prochaine version de la bibliothèque CGAL.

Mots-clés : Analyse en composantes principales.

1 Introduction

Principal Component Analysis (PCA) is a data analysis tool for determining a linear subspace of input data such that the variance of the projection of the data onto the subspace is maximized [Jol02]. More specifically, PCA amounts to compute an orthogonal coordinate system such that the greatest variance of the orthogonal projection of the data lies on the first coordinate (so-called principal component), the second greatest variance lies on the second coordinate, and so on. Equivalently, the linear subspaces spanned by the principal components minimize the sum of squared distances from the data to their projection onto these subspaces. At the intuitive level, PCA provides a way to reduce the dimensionality of complex data so as to reveal the simplified structure that underlies them.

PCA is one of the most popular tool for data analysis, used in applications ranging from statistics to geometric computing through mechanical engineering. For geometric applications, PCA is commonly used either in data space or in a transformed space for dimensionality reduction, normal estimation [MNG04], surface reconstruction [Hop94], surface approximation [CSAD04], shape alignment and matching [WBZ05], inertia tensors [BB04], to cite a few. PCA is mostly used on point set data in statistics and geometric applications, or on polyhedra in mechanics. Although one way to generalize it to arbitrary objects can be achieved through uniform sampling, it is more accurate and often faster to derive closed forms when they exist. In this paper we specialize these derivations to sets of various 2D and 3D geometric primitives, and list potential geometry processing applications.

2 Point Sets

In principal component analysis, least squares analysis is used to determine the required linear subspace. The main idea is to compute the covariance matrix of the points in the point set, perform its eigen decomposition and then determine an *n-dimensional* subspace along the eigenvectors corresponding to the *top n* eigenvalues. Eigenvectors corresponding to large eigenvalues are the directions in which the data has a strong component, or equivalently large variance. We now briefly describe this method.

Let $P = \{p_i\}_{i=1}^n$ be a set of points in 3D and c their center of mass. Consider the same point set P but now centered at the center of mass $\{q_i = p_i - c\}_{i=1}^n$. The covariance matrix of P is defined as:

$$C_P = \frac{1}{n}\sum_{i=1}^{n}\begin{pmatrix} x_i^2 & x_iy_i & x_iz_i \\ x_iy_i & y_i^2 & y_iz_i \\ x_iz_i & y_iz_i & z_i^2 \end{pmatrix} = \frac{1}{n}\sum_{i=1}^{n} q_i q_i^T$$

After an eigendecomposition of C, choosing the eigenvector corresponding to the *top n* eigenvalues gives us n principal components.

Now consider $P = \{s_i\}_{i=1}^n$ where each s_i is a continuous set of points in the space resulting in an object such as a segment, triangle, tetrahedron etc. If x represents the

coordinate vector of points in space relative to the center of mass of the set of objects P, the covariance matrix C_i of each s_i is,

$$C_i = \int_{s_i} xx^T dx$$

For the set P, the covariance matrix is defined as $C = \sum_{i=1}^{n} C_i$. We can then similarly do an eigen-decomposition of C and choose the eigenvector corresponding to the *top* n eigenvalues to give us n principal components. Thus, performing Principal Components Analysis on a set of objects requires calculating the covariance matrix corresponding to each object and using it to calculate the covariance matrix of the object set as a whole. We now describe a method to achieve this goal for a set of geometric objects.

3 Other Geometric Primitives

In this section, we describe the algorithm used for constructing the covariance matrix of any object set. We use a set of 2D triangles as an example to illustrate the algorithm. The notation used is as follows: if u refers to any object or set of objects, M_u is the second order moment of u with respect to the origin; m_u is the mass of u (area in 2D, volume in 3D), c_u is the center of mass of u and C_u is the covariance matrix of u.

For each type of object s, we first construct a canonical geometry of the same class whose covariance matrix can be easily calculated and which can be mapped onto an arbitrary object of this class. We call such a geometry a *standard object*. Suppose o is a standard object corresponding to the class of objects like s. For the class of triangles $s_t = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$, the standard object is $o_t = \{(0,0), (1,0), (0,1)\}$. The second order moment of o_t with respect to the origin is $M_{o_t} = \begin{bmatrix} 1/12 & 1/24 \\ 1/24 & 1/12 \end{bmatrix}$. Next, we find an affine transformation that maps the standard object onto an arbitrary object. Let x_o be the point coordinates on o and x_s, the coordinates of corresponding points on s; we need to find the affine *transformation matrix* A_s and the *offset vector* V_s such that $x_s = A_s x_o + V_s$. Thus, for the class of triangles, $A_t = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}$ and $V_t = [x_1, y_1]^T$. Using this transformation and the second order moment M_o of the standard geometry, we can find the second order moment M_s with respect to the origin of the arbitrary object.

$$\begin{aligned} M_s &= \int_s x_s x_s^T dx_s. \\ &= \int_s (A_s x_o + V_s)(A_s x_o + V_s)^T dx_s \\ &= \int_s (A_s x_o + V_s)(x_o^T A_s^T + V_s^T) dx_s \end{aligned}$$

$$
\begin{aligned}
M_s &= \int_s (A_s x_o x_o^T A_s^T + A_s x_o V_s^T + V_s x_o^T A_s^T + V_s V_s^T) dx_s \\
&= A_s \left(\int_o x_o x_o^T \frac{m_s}{m_o} dx_o \right) A_s^T + \left(\int_s A_s x_o \, dx_s \right) V_s^T \\
&\quad + V_s \left(\int_s (A_s x_o)^T dx_s \right) + V_s V_s^T \left(\int_s dx_s \right) \\
&= \left(\frac{m_s}{m_o} \right) (A_s M_o A_s^T) + \left(\int_s (x_s - V_s) \, dx_s \right) V_s^T \\
&\quad + V_s \left(\int_s (x_s - V_s)^T dx_s \right) + V_s V_s^T m_s \\
&= \left(\frac{m_s}{m_o} \right) (A_s M_o A_s^T) + m_s (\overline{x_s} - V_s) V_s^T + V_s m_s (\overline{x_s^T} - V_s^T) \\
&\quad + V_s V_s^T m_s \text{ where } \overline{x_s} \text{ is the centroid} \\
M_s &= \left(\frac{m_s}{m_o} \right) (A_s M_o A_s^T) + m_s \left(c_s V_s^T + V_s c_s^T - V_s V_s^T \right)
\end{aligned}
$$

For the set of triangles, $m_o = 1/2$ and $m_s = \triangle_s = 1/2 \ |A_t|$; where $\triangle_s$ is its area. Thus, $M_s = |A_t| A_t M_{o_t} A_t^T + \frac{1}{2} |A_t| \left(c_s V_t^T + V_t c_s^T - V_t V_t^T \right)$.
Now consider a set $P = \{s_i\}_{i=1}^n$ of objects.

$$
\begin{aligned}
M_P &= \sum_{i=1}^{n} M_{s_i} \\
m_P &= \sum_{i=1}^{n} m_{s_i} \\
c_P &= \frac{1}{m_P} \left(\sum_{i=1}^{n} m_{s_i} c_{s_i} \right)
\end{aligned}
$$

To calculate the combined covariance matrix C_P,

$$\begin{aligned}
C_P &= \int_P (x_P - \overline{x_P})(x_P - \overline{x_P})^T dx_P. \\
&= \int_P (x_P - \overline{x_P})(x_P^T - \overline{x_P}^T) dx_P. \\
&= \int_P x_P x_P^T - \overline{x_P} x_P^T - x_P \overline{x_P}^T + \overline{x_P x_P}^T dx_P. \\
&= M_P - \overline{x_P} \int_P x_P^T dx_P - \int_P x_P dx_P \, \overline{x_P}^T \\
&\qquad + \overline{x_P x_P}^T \int_P dx_P. \\
&= M_P - m_P \, \overline{x_P} \, \overline{x_P}^T. \\
C_P &= M_P - m_P \, c_P \, c_P^T.
\end{aligned}$$

Thus, the closed form formula for the covariance matrix of a set of objects is given by,

$$\begin{aligned}
C_P = \sum_{i=1}^{n} \left[\left(\frac{m_{s_i}}{m_o} \right) (A_{s_i} M_o A_{s_i}^T) + m_{s_i} \left(c_{s_i} V_{s_i}^T + V_{s_i} c_{s_i}^T - V_{s_i} V_{s_i}^T \right) \right] \\
- \frac{1}{(\sum_{i=1}^{n} m_{s_i})} \left(\sum_{i=1}^{n} m_{s_i} c_{s_i} \right) \left(\sum_{i=1}^{n} m_{s_i} c_{s_i}^T \right)
\end{aligned}$$

The covariance formula above is general and applies to arbitrary sets of primitives. Appendix A and B show several standard objects and their associated order-2 moment w.r.t. origin, transformation matrix and offset vectors. It is interesting to note that when the union of all primitives can be decomposed into a set of primitives sharing the same base point as center of mass, the formula simplifies to the sole affine transform part. Let c_P be the center of mass of the union of all primitive objects in the set such that all of them have a vertex at c_P. Translate the axes such that the origin is at c_P. Thus, the standard object o' with respect to the new origin is given by $x_{o'} = x_o + c_p$ and its second order moment w.r.t c_P is $M_{o'} = M_o$. Now, we just need to scale o' to fit all objects in the set. Thus, we can find a matrix A such that $x_s = A_s x_{o'}$. So, the covariance of the set P is given by

$$C_P = \sum_{i=1}^{n} \left(\frac{m_{s_i}}{m_o} \right) A_{s_i} M_o A_{s_i}^T$$

A concrete example use of this formula has been used in [ACSTD07] to calculate the covariance matrix of unions of Voronoi cells for surface normal estimation from noisy point sets. Consider a bounded Voronoi cell V. The covariance matrix of a Voronoi Cell can be calculated by first computing its center of mass o and decomposing it into tetrahedra. The

decomposition is performed in a star fashion such that all tetrahedra t_i share the same base point which coincides with o. This decomposition is always possible as the Voronoi cell is convex.

Since all tetrahedra consist of a common vertex at the center of mass of the voronoi

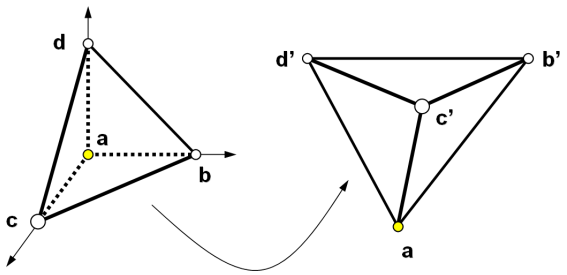


Figure 1: Affine Transform of a standard tetrahedron to an arbitrary one

cell, we can transform the origin to the center of mass. For a single arbitrary tetrahedron $s = (a, b', c', d')$, $o = (a, b, c, d)$ is the standard geometry. Note that s and o have a common vertex a which is the center of mass of the voronoi cell. For an affine transformation to map o onto s, the transformation matrix A_s is defined as:

$$A_s = \begin{pmatrix} (b'-a').x & (c'-a').x & (d'-a').x \\ (b'-a').y & (c'-a').y & (d'-a').y \\ (b'-a').z & (c'-a').z & (d'-a').z \end{pmatrix}$$

The order-2 moment matrix M_o of the canonical standard tetrahedron o with respect to the origin is:

$$\begin{aligned} M_o &= \int_{x=0}^{1} \int_{y=0}^{1-x} \int_{z=0}^{1-x-y} \begin{pmatrix} x^2 & xy & xz \\ yx & y^2 & yz \\ zx & zy & z^2 \end{pmatrix} dzdydx \\ &= \frac{1}{120} \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \end{aligned}$$

The mass of s is $m_s = |A_s|/6$ and the mass of o is $m_o = 1/6$. For the voronoi cell V decomposed into a set of tetrahedra $V = \{s_i\}_{i=1}^{n}$, its covariance matrix C_P thus is,

$$C_P = \sum_{i=1}^{n} |A_{s_i}| A_{s_i} M_o A_{s_i}^T$$

4 Implementation

The algorithm described in this paper is generic and can be applied to various types of geometries. This power will be available in the next release of the CGAL library [c] for most finite primitives of the 2D and 3D kernel: 2D and 3D Points, 2D and 3D segments, 2D and 3D triangles, 2D rectangles, 3D cuboids, 3D tetrahedra, 2D circles and 3D spheres.

The API follows the STL paradigm. The user must at least provide an iterator range of primitives, and the type of linear subspace to be fitted (2D line, 3D line, or 3D plane). The function returns a scalar $\in [0;1]$ indicating the quality of the fit related to ratios of the covariance eigenvalues (0 for isotropic data sets, 1 for zero error fit). For example one can fit a line to a 2D point set by calling the function CGAL::linear_least_squares_fitting_2(-points.begin(),points.end(),line) or a 3D line to a set of tetrahedra by calling CGAL::linear_least_squares_fitting_3(tetrahedra.begin(),tetrahedra.end(),line). or a 3D plane to a set of spheres by calling CGAL::linear_least_squares_fitting_3(spheres.begin(),spheres.end(),-plane). The user can fit a point as well by calling CGAL::centroid(begin,end).

Furthermore, the user can provide an additional optional argument specifying the dimension of the primitive to be considered. By default, this argument is the dimension of the primitives in the container (0 for points, 1 for segments, 2 for triangles, rectangles and spheres, 3 for tetrahedra and cuboids). The user can, for e.g., fit a linear subspace to a set of balls (instead of spheres) by specifying 3 as dimension, or can fit the edges of a set of tetrahedra by specifying 1 as dimension, or the boundary triangles of a set of tetrahedra by specifying 2, without having to fill another container of primitives. For cases where the user wants to fit more than just one primitive we also propose the functions std::pair<Line,Plane> CGAL::least_squares_line_plane_fitting_3(begin,end) and CGAL::Triple<Point,Line,Plane> CGAL::least_squares_point_line_plane_fitting_3(begin,end)

5 Applications

From the general application point of view, the main added value of the present geometric computations becomes relevant when the discretization of a shape is provided by more complex primitives than points, such as segments, triangles, tetrahedra, etc. The proposed closed forms are more accurate and often more efficient than the common uniform point sampling followed by point-based PCA.

Example potential applications are shape alignment (i.e., finding canonical frames) and computation of moment of inertia for surface triangle meshes, without having to uniformly point sample the mesh. The same would be valid for embedded graphs where the primitives would be 2D or 3D segments for the graph edges, or for solids once properly discretized into tetrahedra. For various data analysis applications, a common approach consists of carrying on PCA computations on point sets living in a so-called feature space. Instead of weighting

the points as commonly done, one could replace these weighted points by balls.

One recent trend consists of elaborating upon integral approaches for geometric data analysis and processing (see e.g., [PWY*07]), in order to reduce the dependence to the input discretization. This work can be seen as one step further in the same direction.

6 Conclusion

This work proposes closed-form computations of covariance matrices of sets of various 2D and 3D geometric primitives, together with a generic implementation for the next release of the CGAL library. The main added value is to remove the need for the user to point sample a shape in order to perform approximate PCA computations, and to perform direct computations over more elaborate discretizations.

As future work we plan to extend these closed forms to a richer set of primitives, such as sets of anisotropic versions of the current handled primitives or primitives in higher dimension. We plan to enrich the API so as to handle containers of mixed primitives with homogeneous dimensions, so as to fit linear subspaces e.g. to sets of tetrahedra, cuboids and balls. Another extension is to handle weighting functions, such as radial Gaussian kernels used, e.g., for normal estimation or MLS surface fitting. Finally, we want to extend this work to unions of primitives, as well as to arbitrary Boolean operations over sets of primitives.

References

[ACSTD07] Alliez P., Cohen-Steiner D., Tong Y., Desbrun M.: Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Fifth EUROGRAPHICS Symposium on Geometry Processing* (2007), Belyaev A. G., Garland M., (Eds.), vol. 257, pp. 39–48.

[BB04] Blow J., Binstock A. J.: How to find the inertia tensor (or other mass properties) of a 3d solid body represented by a triangle mesh, 2004.

[c] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. *ACM Trans. Graph 23*, 3 (2004), 905–914.

[Hop94] Hoppe H.: *Surface Reconstruction from Unorganized Points*. PhD thesis, Dept. of Computer Science and Engineering, U. of Washington, 1994.

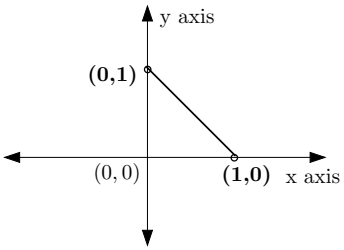
[Jol02] Jolliffe I. T.: *Principal Component Analysis*. Springer-Verlag, New York, 2002.

[MNG04] Mitra N. J., Nguyen A., Guibas L.: Estimating surface normals in noisy point cloud data. In *special issue of International Journal of Computational Geometry and Applications* (2004), vol. 14 (4–5), pp. 261–276.

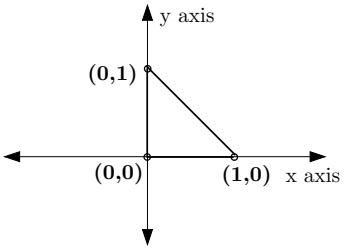
[PWY*07] Pottmann H., Wallner J., Yang Y.-L., Lai Y.-K., Hu S.-M.: Principal curvatures from the integral invariant viewpoint. *Comput. Aided Geom. Design 24* (2007), 428–442.

[WBZ05] Wang B., Bangham J. A., Zhu Y.: Shape retrieval by principal components descriptor. In *Pattern Recognition and Image Analysis, Third International Conference on Advances in Pattern Recognition 2005* (2005), Singh S., Singh M., Apté C., Perner P., (Eds.), vol. 3687, Springer, pp. 626–634.

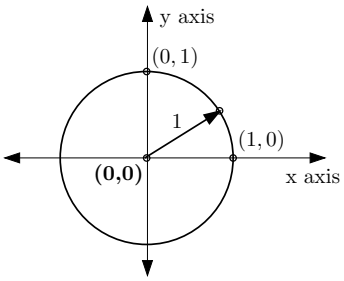
A Standard Geometries of Various Shapes



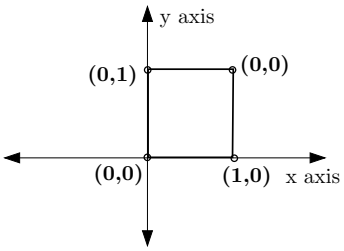
Std. 2D Segment: $\{(1,0),(0,1)\}$.



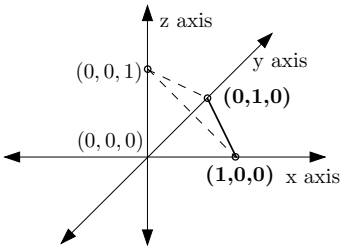
Std. 2D Triangle: $\{(0,0),(1,0),(0,1)\}$.



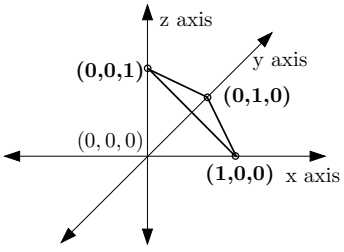
Std. 2D Circle/Disk:$\{center = (0,0), r = 1\}$



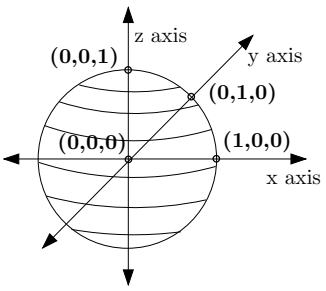
Std. 2D Axis-Aligned Rectangle: $\{(0,0),(1,1)\}$.



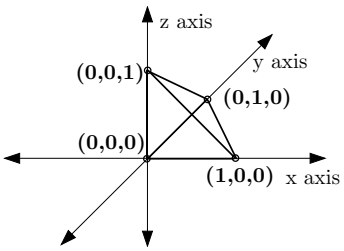
Std. 3D Segment: $\{(1,0,0),(0,1,0)\}$.



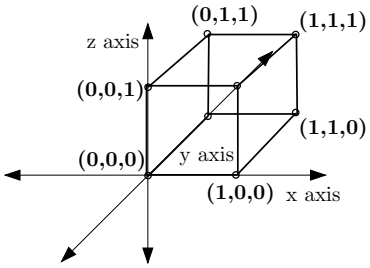
Std. 3D Triangle: $\{(1,0,0),(0,1,0),(0,0,1)\}$.



Std. 3D Sphere: $\{center = (0,0,0), r = 1\}$.



Std. 3D Tet:$\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\}$.



Std. 3D Cuboid: $\{(0,0,0),(1,1,1)\}$.

B Data Sheet

Sr. No.	Shape	Order-2 Moment of Standard Object about origin	Transformation Matrix	Offset Vector
1	2D Segments $s_i = \{(x_1, y_1), (x_2, y_2)\}$	$M = \begin{bmatrix} \sqrt{2}/3 & \sqrt{2}/6 \\ \sqrt{2}/6 & \sqrt{2}/3 \end{bmatrix}$	$A = \begin{bmatrix} x_1 & x_2 \\ y_1 & y_2 \end{bmatrix}$	$V = [0, 0]^T$
2	2D Triangle $t_i = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$	$M = \begin{bmatrix} 1/12 & 1/24 \\ 1/24 & 1/12 \end{bmatrix}$	$A = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}$	$V = [x_1, y_1]^T$
3	2D Disk $c_i = \{center = (x_c, y_c), radius = r\}$	$M = \begin{bmatrix} \pi/4 & 0 \\ 0 & \pi/4 \end{bmatrix}$	$A = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$	$V = [x_c, y_c]^T$
4	2D Circle $c_i = \{center = (x_c, y_c), radius = r\}$	$M = \begin{bmatrix} \pi & 0 \\ 0 & \pi \end{bmatrix}$	$A = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix}$	$V = [x_c, y_c]^T$
5	2D axis-aligned Rectangle $r_i = \{(x_1, y_1), (x_2, y_2)\}$ $[x_1, y_1]^T$ = left-bottom corner $[x_2, y_2]^T$ = right-top corner	$M = \begin{bmatrix} 1/3 & 1/4 \\ 1/4 & 1/3 \end{bmatrix}$	$A = \begin{bmatrix} x_2 - x_1 & 0 \\ 0 & y_2 - y_1 \end{bmatrix}$	$V = [x_1, y_1]^T$
6	3D Segments $s_i = \{(x_1, y_1, z_1), (x_2, y_2, z_2)\}$	$M = \begin{bmatrix} \sqrt{2}/3 & \sqrt{2}/6 & 0 \\ \sqrt{2}/6 & \sqrt{2}/3 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$A = \begin{bmatrix} x_1 & x_2 & 0 \\ y_1 & y_2 & 0 \\ z_1 & z_2 & 1 \end{bmatrix}$	$V = [0, 0, 0]^T$
7	3D Triangle $t_i = \{(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)\}$	$M = \begin{bmatrix} 1/12 & 1/24 & 1/24 \\ 1/24 & 1/24 & 1/12 \\ 1/24 & 1/24 & 1/12 \end{bmatrix}$	$A = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix}$	$V = [0, 0, 0]^T$
8	3D Solid Sphere $c_i = \{center = (x_c, y_c, z_c), radius = r\}$	$M = \begin{bmatrix} 4\pi/15 & 0 & 0 \\ 0 & 4\pi/15 & 0 \\ 0 & 0 & 4\pi/15 \end{bmatrix}$	$A = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix}$	$V = [x_c, y_c, z_c]^T$
9	3D Tetrahedron $t_i = \{(x_1, y_1, z_1), \ldots, (x_4, y_4, z_4)\}$	$M = \begin{bmatrix} 1/60 & 1/120 & 1/120 \\ 1/120 & 1/60 & 1/120 \\ 1/120 & 1/120 & 1/60 \end{bmatrix}$	$A = \begin{bmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{bmatrix}$	$V = [x_1, y_1, z_1]^T$
10	3D axis-aligned Solid Cuboid $r_i = \{(x_1, y_1, z_1), (x_2, y_2, z_2)\}$ $[x_1, y_1, z_1]^T$,$[x_2, y_2, z_2]^T$ = closest, farthest diagonal corners	$M = \begin{bmatrix} 1/3 & 1/4 & 1/4 \\ 1/4 & 1/3 & 1/4 \\ 1/4 & 1/4 & 1/3 \end{bmatrix}$	$A = \begin{bmatrix} x_2 - x_1 & 0 & 0 \\ 0 & y_2 - y_1 & 0 \\ 0 & 0 & z_2 - z_1 \end{bmatrix}$	$V = [x_1, y_1, z_1]^T$
11	3D axis-aligned Hollow Cuboid $r_i = \{(x_1, y_1, z_1), (x_2, y_2, z_2)\}$ $[x_1, y_1, z_1]^T$,$[x_2, y_2, z_2]^T$ = closest, farthest diagonal corners	$M = \begin{bmatrix} 7/3 & 3/2 & 3/2 \\ 3/2 & 7/3 & 3/2 \\ 3/2 & 3/2 & 7/3 \end{bmatrix}$	$A = \begin{bmatrix} x_2 - x_1 & 0 & 0 \\ 0 & y_2 - y_1 & 0 \\ 0 & 0 & z_2 - z_1 \end{bmatrix}$	$V + [x_1, y_1, z_1]^T$

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399